METHOD AND SYSTEM FOR CONTROLLING DEFAULT VALUES OF FLIP-FLOPS IN PGA/ASIC-BASED DESIGNS

FIELD OF THE INVENTION

[0001] This invention relates generally to programmable logic devices, and more particularly to a method and system for preloading default values in flip-flops (FFs), which can be utilized as configuration registers, in PGA/ASIC-based designs.

BACKGROUND OF THE INVENTION

[0002] As chip capacity continues to significantly increase, the use of programmable gate arrays (PGAs), particularly field programmable gate arrays (FPGAs), is quickly replacing the use of application specific integrated circuits (ASICs). An ASIC is a specialized chip that is designed for a particular application. Notably, an FPGA is a programmable logic device (PLD) that has an extremely high density of electronic gates as compared to an ASIC. This high gate density has contributed immensely to the popularity and flexibility of FPGA's. Importantly, FPGAs can be designed using a variety of architectures, which can include user configurable input/output blocks (IOBs) and programmable/configurable logic blocks (PLBs/CLBs) having configurable interconnects and switching capability. The CLBs, IOBs, and interconnect structure are typically programmed by loading a bitstream of configuration data into internal configuration memory cells that define how the CLBs, IOBs, and interconnect structure are configured. The configuration bitstream is typically loaded from an external memory, for example, from and an external programmable read only memory (PROM) or similar device. The collective states

of the individual memory cells determine the functions of the FPGA.

[0003] As advancements in computer chip technology continue to increase the threshold on the number of CLBs, IOBs and interconnect structure in FPGAs, it has become possible to build entire data processing systems inside the FPGA. These processing systems are typically called embedded processors or controllers. An embedded processor or controller can be microprocessor or microcontroller circuitry that has been integrated into an electronic device, such as an FPGA, as opposed to being built as a standalone module or "plugin card." Moreover, advancements in FPGA technology has also led to the development of FPGA-based system-on-chip (SoC), including FPGA-based embedded processor SoCs. A SoC is a fully functional product having its electronic circuitry contained on a single chip. While a microprocessor chip requires ancillary hardware electronic components to process instructions, a SoC can include all required ancillary electronics. For example, a SoC for a cellular telephone can include a microprocessor, encoder, decoder, digital signal processor (DSP), RAM and ROM. FPGA-based SoCs with embedded operating systems (OSs) have further enhanced their popularity and flexibility.

[0004] FPGA-based SoCs have resulted in the proliferation of numerous consumer devices such as wireless telephones, personal digital assistants (PDAs), and digital cameras. In order for device manufacturers to develop FPGA-based SoCs, it is necessary for them to acquire intellectual property rights for system components and/or related technologies that are utilized to create the FPGA-based SoCs. These system components and/or technologies are called cores or IP cores. An electronic file containing component information can typically be used to represent the core. A device

manufacturer will generally acquire rights for one or more IP cores that are integrated to fabricate the SoC.

[0005] One of the most important resources in data processing systems including consumer devices is memory. Many FPGAs provide blocks of random access memories (RAMs), each having thousands of memory cells (called "block RAMs" or BRAMS). BRAMS are known in the art. Notwithstanding, the flexibility of the blocks permit a variety of memory configurations. For example, a BRAM having a capacity of 16 Kilobits can be configured to have an address depth of either 16K, 8K, 4K, 2K, 1K and 0.5K, with the corresponding number of bits per address as 1, 2, 4, 8, 16 or 32, respectively. Additionally, a number of blocks can be combined to increase the total memory size. During operation of the FPGA, the BRAMs can be used to store information such as system configuration data, program data, and operational data. For example, configuration information used during initialization of a data processing device can be stored in the BRAM.

[0006] The high level structure of an FPGA comprises the FPGA fabric, which can include a processor block, and BRAMs. The processor block can include an IP core surrounded by supporting logic circuitry, known in the art as "gasket logic." Residing within the gasket logic are on-chip memory (OCM) controllers configured to control the flow of data between the BRAMS and the IP core. The OCM controllers typically have control registers which store configuration information, which can be used for initialization and performing particular functions.

[0007] One major disadvantage with existing data processing systems and devices that utilize FPGAs is that the on-chip devices such as the OCM controllers have to wait for the system or device to be initialized before the control registers are configured. Generally, the on-chip device

control or configuration registers are loaded at runtime, which occurs after the data processing system or device has been powered up and has executed its startup routines. The more extensive the startup routines, the longer it will take to load values in the control registers during system initialization. Consequently, boot and startup times for these data processing systems and devices can be unnecessarily extended.

[0008] Given these and other inherent drawbacks, there is a need for a method and system for controlling default value of flip-flops used as configuration registers in PGA/ASIC-based designs.

SUMMARY OF THE INVENTION

[0009] A method and system for pre-configuring an FPGA to a known state prior to the FPGA undergoing system initialization or a reset condition of an IP core of the FPGA is shown. The method can include the step of loading a configuration value for an FPGA on-chip device into a flip-flop, which can be coupled to the on-chip device. The configuration value can subsequently be transferred or read from the flip-flop to the on-chip device in order to effectuate pre-configuration of the on-chip device. The configuration value can be pre-stored in an FPGA memory cell, which can be a BRAM memory cell. The configuration value can be transferred from the memory cell to the flip-flop upon power-up of the FPGA or upon a reset condition of the IP core. A transitioning clock signal coupled to the flip-flop can effectuate the transfer of the configuration value from the memory cell to the flip-flop upon power-up of the FPGA (or the reset of the IP core). The transitioning clock signal can also effectuate the transfer or reading of the configuration value from the flip-flop to the on-chip device.

The flip-flop can be a D-type flip-flop, which can function as a configuration register for the on-chip device. The on-chip device can be an on-chip memory controller.

[0010] In another aspect of the invention, a method for pre-configuring an FPGA prior to system initialization of the FPGA (or during a reset condition of the IP core) comprises pre-configuring the FPGA by loading a configuration value into at least one flip-flop of the FPGA that function as a configuration register for a memory controller of the FPGA. The value in the flip-flop can be used to configure the FPGA to a specified state. Loading the configuration value into the FPGA can be accomplished by transferring a value previously stored in a memory cell of the FPGA to a flip-flop where it can be stored. Whenever the FPGA is powered up (or the IP core receiving a reset signal to reset the internal states of the IP core), the value stored in the flip-flop can be read from the flip-flop in order to effectuate configuration of the on-chip device. The value stored in the memory cell can be a default state of the memory cell.

[0011] In yet another aspect of the present invention, a system for pre-configuring an FPGA to a known state prior to the system initialization of the FPGA comprises a flip-flop coupled to an on-chip device of the FPGA and a clock transitioning circuit coupled to the flip-flop. The clock transitioning circuit can cause a configuration value for the on-chip device to be loaded into the flip-flop. The clock transitioning circuit can further cause the configuration value to be read by the on-chip device, thereby effectuating pre-configuration of the on-chip device. The configuration value can be pre-stored in an FPGA BRAM memory cell. Furthermore, the flip-flop can be a D-type flip-flop, which can be configured to operate as a configuration register for the on-chip device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 is a block diagram of an exemplary FPGA-based system.

[0013] FIG. 2 is a block diagram of an exemplary OCM controller 106 in accordance with the FPGA-based system of FIG. 1.

[0014] FIG. 3 is an exemplary block diagram of a system for controlling default values of flip-flops in accordance with the inventive arrangements.

DETAILED DESCRIPTION OF THE DRAWINGS

[0015] Referring to FIG. 1, there is shown a block diagram of an exemplary FPGA-based system 100. Referring to FIG. 1, the FPGA-based system 100 includes the FPGA fabric 102. The FPGA fabric 102 can include a processor block 104 and a BRAM segment 120. The BRAM segment 120 can include one or more BRAMS, each BRAM comprised of a plurality of memory cells. Memory cell 108 is a representative memory cell of a BRAM 122 within BRAM segment 120. However, the invention is not limited in this regard and there can be one or more BRAM segments within the FPGA Fabric 102 or other memory configurations.

[0016] The processor block 104 can include an IP core 110 and an on-chip memory (OCM) controller 106. Exemplary IP core 110 can include, but is not limited to, a microprocessor core or microcontroller core. The IP core 110 can be coupled to OCM controller 106 through a shared control bus 112. OCM controllers such as OCM controller 106 are known in the art. OCM controller 106 can be configured to control the communication of data between the BRAM segment 120 and the IP core 110.

[0017] FIG. 2 is a block diagram of the exemplary OCM controller 106 in accordance with the FPGA-based system of FIG. 1. Referring to FIG. 2, OCM controller 106 can include a control block 212, a wait state manager (WSM) 214, and an address manager (AM) 216. Control block 212 preferably interfaces with IP core 110 and BRAM segment 120. The wait state manager 214 can be configured to generate wait states used for operation of control block 212 in accordance with any combination of user and IP core 110 requirements. The address manager 216 can be configured to setup the address depth of the BRAM segment 120 in accordance with user requirements.

[0018] Wait state manager 214 can be configured to communicate with register 224 located within IP core 110, such communication preferably occurring via control bus 112. Additionally, the wait state manager 214 can be configured to communicate with memory 222 which can be part of processor block 104 (see FIG. 1) or BRAM segment 120. Notwithstanding, memory 222 may comprise one or more memory cells, depending on the amount of data to be stored therein. The size of register 224 can also depend on the number of wait states to be generated. The wait state manager 214 can also be configured to control the operation of control block 212 by directing control block 212 to generate wait states corresponding to the data contents of register 224 and/or memory 222.

[0019] Address manager 216 can be configured to accept inputs from register 228 located within IP core 110 via address bus 210. Additionally, the address manager 216 can be configured to communicate with memory 226, which can be part of processor block 104 (See FIG. 1) or the BRAM segment 120. Notwithstanding, memory 226 can include one or more memory cells, depending on the amount of data to be stored

therein. In general, the size of register 226 can be dependent on the number of bits required to address the BRAM segment 120.

[0020] The address manager 216 can permit optimal access to BRAM segment 120. Particularly in a case where the BRAM 120 is divided into separate blocks, the address manager 216 can be permitted prioritized access to specified BRAM segments. For example, the BRAM can be divided into global BRAM and local BRAM. In this case, the address manager 216 can be configured to have prioritized access to either the local BRAM or the global BRAM to minimize access delay.

[0021] Control block 212 preferably interfaces with a memory management unit (MMU) 232 of IP core 110. The MMU 232 can be configured to generate read address signals for data load and instruction fetch operations. Additionally, MMU 232 can also be configured to generate write address signals for data to be written into memory. The MMU 232 can generate and receive appropriate control signals from control block 212 to facilitate certain operations. For example, control block 212 can accept read and write data bus signals from MMU 232. The control block 212 can further be configured to deliver read and write data signals to BRAMs 234 and 236, each BRAM preferably comprising one or more memory cells 108. Notwithstanding, it should be recognized that FIG. 2 is an exemplary block diagram and the invention in not limited in this regard. Pending United States patent application, Serial Number 09/917,30, Xilinx Inc., the assignee, entitled "User Configurable Memory System having Global Memory Blocks," provides a more exhaustive description of an exemplary OCM controller.

[0022] FIG. 3 is an exemplary block diagram of a system for controlling default values of flip-flops in accordance with the inventive arrangements. A basic configuration

register can be a single bit register capable of implementing a binary operation. In this regard, a simple 1-bit configuration register can be implemented by a single D type flip-flop (D-FF). Flip-flops are well known in the art and form the basic building blocks of most digital circuits and integrated circuits (IC). Advantageously, a single D-FF can be used as a configuration register to store a value for controlling the operation of OCM controller 106 prior to booting and for software controlled configuration of the FPGA 102. The sequence of events for FPGA start up comprises the powering up of the FPGA, a reset signal being asserted, the FPGA being configured, the reset signal being de-asserted (when the configuration is complete), and the loading of boot or startup software. Thus, configuration of the FPGA occurs during a reset phase as will become more apparent in the description below.

[0023] Referring to FIG. 3, the OCM controller 106 can include a D-FF 312, and multiplexers 316 and 318. The D-FF 312 comprises inputs "D" and clock (CLK) signal 314. The D input of the D-FF 312 can be connected to a particular memory cell of BRAM segment 120. Upon FPGA 102 being powered and during reset phase, in advance of FPGA 102 being configured by boot or startup software, a pre-stored value in memory cell 108 can be loaded into the D-FF 312. The loading of the pre-stored value from the memory cell 108 into the D-FF 312 can occur on the transition of the CLK signal 314, since D-FF is an edge triggered device. Additionally, to accomplish loading of the D-FF 312 with the value from the memory cell 108, the reset (RST) signal 322 on the multiplexer 318 is held high (1). When the reset phase is done, the RST signal 322 is set to low (0). At this point the D-FF 312 representing the 1-bit control register is loaded.

[0024] Since the output Q of the D-FF 312 follows the D input of D-FF 312, D-FF 312 will remain in its loaded state since the Q output of D-FF 312 is looped back through multiplexers 316 and 318 respectively, into the D input of D-FF 312. This ensures that the D-FF 312 retains its loaded value until the value is purposely changed. The OCM controller 106 can use the value loaded in the D-FF 312 to control its operation prior to being initialized by any system or device software. Although an edge triggered D-FF is illustrated, the invention is not limited in this regard. Furthermore, the negated Q output of the D-FF is not shown, since it is not relevant to the understanding and practice of the invention.

[0025] Subsequent to loading system software, the value loaded in the control register or 1-bit D-FF 312 can be overwritten by the system software or application. In this case, the IP core 110 under the control of system software or an application, can write to a device control register. This device control register can be memory mapped to the physical memory address of D-FF 312. Importantly, to overwrite the value in the D-FF 312, the write enable signal (WrEn) 324 of multiplexer 316 can be held high (1) by IP core 110 via bus 112. Although the needed signals for asserting WrEn 324 come via bus 112, decoding logic (not shown) in OCM Controller 106 can be used to decode those needed signals to assert WrEn 324. Subsequent to loading D-FF 312, the WrEn 324 can be held low (0). Importantly, loop back path 320 ensures that once data stored in D-FF 312, the value will remain in the D-FF 312 until the RST signal 322 or the WrEn signal 324 is pulled high (1).

[0026] Although a single D-FF 312 is used to implement a simple configuration register in FIG. 3, it should be recognized by those skilled in the art that the invention is

not limited in this regard. A plurality of D-FFs similar to D-FF 312 can be arranged to form a control register of any required size. Such arrangements are well known in the art. For example, four D-FFs could be arranged to implement a 4-bit configuration register for the OCM controller 106. In operation, the 4-bit configuration register can be used to inform the OCM controller 106 of the number of wait states required to access a particular device, for example BRAM 236 (FIG. 2). In this case, the 4-bit configuration can be used to represent any of 0 through 15 wait states. Similarly, the invention is not limited to configuration of an on-chip memory controller. In this regard, one or more flip-flops can be configured in accordance with the inventive arrangements to function as control register(s) for any FPGA on-chip device.

[0027] In light of the foregoing description of the invention, it should be recognized that the present invention can be realized in hardware, software, or a combination of hardware and software. A method and system for controlling default values of flip-flops in PGA/ASIC-based designs such as FPGA-based embedded processor SoCs according to the present invention can be realized in a centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system, or other apparatus adapted for carrying out the methods described herein, is suited. A typical combination of hardware and software could be a general-purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein.

[0028] The present invention can also be embedded in a computer program product, which comprises all the features

enabling the implementation of the methods described herein, and which, when loaded in a computer system, is able to carry out these methods. Computer program or application in the present context means any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following a) conversion to another language, code or notation; b) reproduction in a different material form.

[0029] Additionally, the description above is intended by way of example only and is not intended to limit the present invention in any way, except as set forth in the following claims.